Accredited Standards Committee
NCITS, Information Technology*
NCITS/L3, Audio/Picture Coding
NCITS/L3.2, Still Image Coding

Document: **NCITS/L3.2/01-015**
Date: Jan. 31, 2001
Project: JPEG-2000

Reply to:
Bernie Brower, Chair
Eastman Kodak Company
1447 St. Paul Street
Rochester, New York 14653-7208
(1 716) 253-5293, brower@kodak.com

To: NCITS/L3.2

From: Chris Brislawn, Brendt Wohlberg, and Michelle Pal, Los Alamos National Lab

Title: Contribution to USNB Comments on Annex G and Annex I, JPEG-2000 Part 2 FCD

USNB Comments on Annex G: Transformation of images, extensions

Add the usual boilerplate “Flowcharts and tables are normative only in the sense that they define....”

G.2, Table G-1.
The parameter beta_s needs to be signed to ensure compatibility with Part 1 (for the 5-3 filter bank).
The parameter sizes should indicate the exact number of bits needed rather than the minimum number of bytes required.

G.3.1.
Considerably more exposition is needed here.

G.3.2, G.3.3.
These sections contain errors and should be expressed in a form more consistent with section G.3.1.

G.4.2.
More detail needed.

G.4.3.2.
Hard to follow these expressions; a bit more exposition needed.

G.4.4.1, G.5.3.1.
There is a significant problem (recently discovered) affecting reversible filtering with HS filter banks that is related to the use of symmetric extension. The VM implements half-sample symmetric extension for lifting by interleaving many short extensions between lifting steps, but if one implements symmetric extension for HS filter banks as a preprocessing step followed by a cascade of lifting steps (as described in Annex G) then reversibility can fail even though the filter bank is integer-to-integer. This problem results from using the floor function for rounding rather than using an odd function like integer-part truncation. Several possible solutions would involve using multiple different rounding rules on different lifting steps (with concomitant signaling). The solution recommended by the USNB Coding Subgroup, however, is to fix the interleaved extension policy already employed in the VM so that it enables reversible coding with arbitrary integer-to-integer HS filter banks with the proviso that it must remain equivalent to pre-extension (i.e., equivalent to the extension policies defined in Part 1) for WS filter banks.

G.7.
Many examples are still missing.

G.8.
Needs changes to explain recommended extension policy and reversible implementation.

Comments on JPEG-2000 Part 2, Annex I, FCD

Summary.

This document constitutes feedback to the USNB on the text of Part 2 Annex I as found in document number WG1N2000. We point out what appear to be a couple of technical mistakes, suggest corrections, and comment on other problems with the current exposition in Annex I. All figure and equation references contained herein refer to the cross-referencing found in document WG1N2000. The issues raised are the following:

1. In general, the exposition too often implicitly assumes that the marker segment syntax defined in Annex A constitutes a self-contained procedural definition. There is a need for much more explicit presentation of procedure flow and procedural rules.
2. Inconsistent use of linear algebra in the definition of the linear dependency removal transform.
3. There is a need for a more detailed, explicit definition of MCC component collections and MIC component collections since these define the input vectors to the dependency removal and decorrelation transforms.
4. Informative subsections on the forward component transform would be very helpful, both in general and for the worked example.

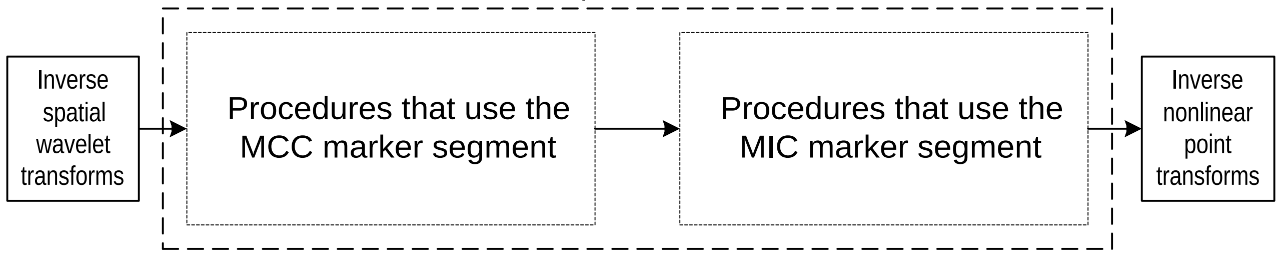
1. Procedures vs. marker segments.

The FCD contains numerous statements of the form "The MCC marker segment may generate null output components as it reorders intermediate components...", which make it appear that "marker segments" are also (apparently self-explanatory) procedures. We recommend removing all such language and replacing it with definitions of procedures (corresponding to blocks in flowcharts) that reference to marker segments for necessary parameters. This style of exposition would be much more consistent with the presentation in the rest of the standard.

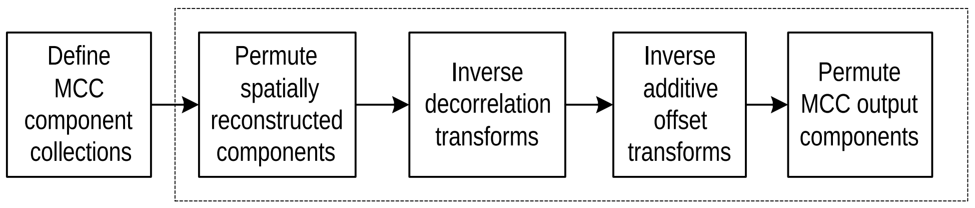
Also on the topic of semantics, it is preferable to eliminate all references to a "spectral" axis in the data (except for the worked example); the standard should refer generically to a "component" direction without making assumptions as to whether it represents a spectral direction, a third spatial dimension, etc.

Component-collection reordering and insertion/deletion operations should be indicated explicitly in the system block diagram. We present a proposed revision of the high-level block diagrams below. The high-level block diagrams contain explicit procedures for permuting components, inserting or deleting components (when appropriate), and grouping components into component collections for input to the various transforms occuring in the process. These procedures all need definitions spelled out in the text.

Inverse component transformation

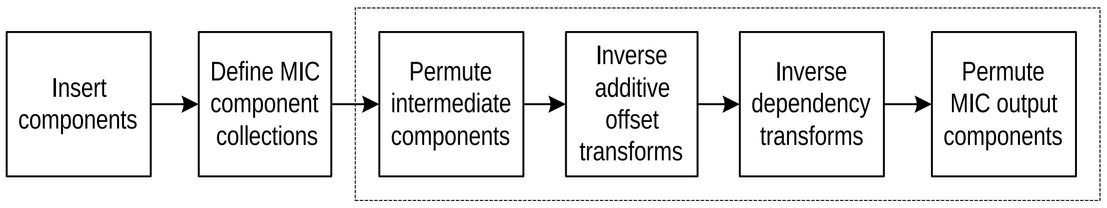

Inverse spatial wavelet transforms
Procedures that use the MCC marker segment
Procedures that use the MIC marker segment
Inverse nonlinear point transforms

Procedures that use the
MCC marker segment :


Define MCC component collections
Permute spatially reconstructed components
Inverse decorrelation transforms
Inverse additive offset transforms
Permute MCC output components

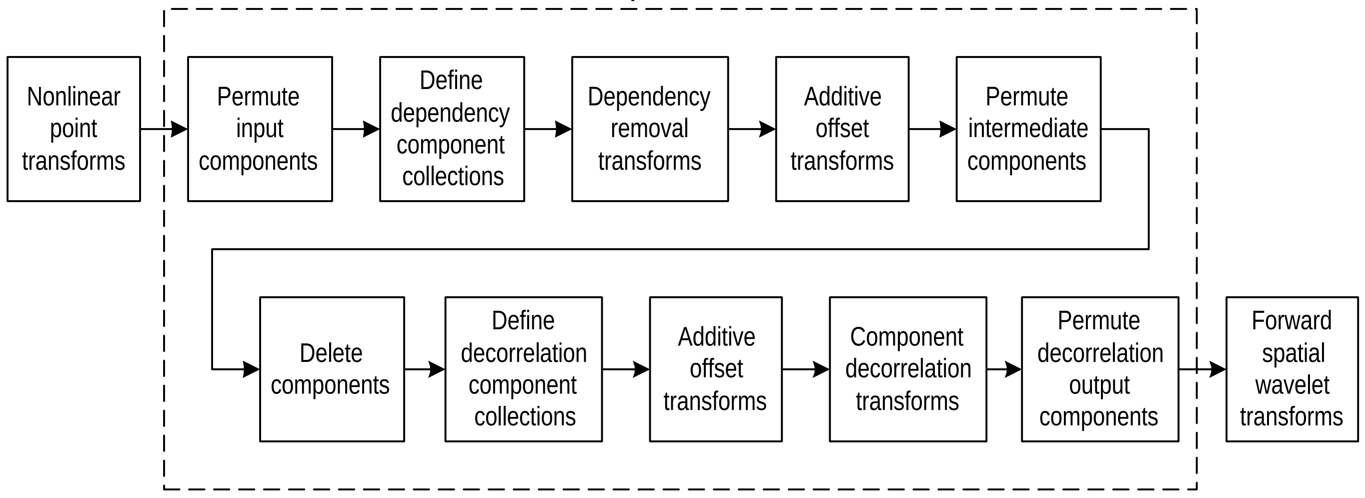
Procedures in the dotted box are
repeated for each component collection

Procedures that use the
MIC marker segment :


Insert components
Define MIC component collections
Permute intermediate components
Inverse additive offset transforms
Inverse dependency transforms
Permute MIC output components

Procedures in the dotted box are
repeated for each component collection

Forward component transformation


Nonlinear point transforms
Permute input components
Define dependency component collections
Dependency removal transforms
Additive offset transforms
Permute intermediate components
Delete components
Define decorrelation component collections
Additive offset transforms
Component decorrelation transforms
Permute decorrelation output components
Forward spatial wavelet transforms

2. Linear Algebra Problem in Equation I.4.

A significant problem with equation I.4 is that the indicated matrix-vector product, $\mathbf{R}\underline{Y}'$, in I.4 does **not** mathematically represent the recursive definition of $\underline{Y}'$ intended by equation I.3. We feel that it is extremely dangerous to present normative definitions in what appears to be matrix-vector product form when in fact this is inconsistent with the intended definition (i.e., equation I.3). A matrix-vector expression of the form $\underline{Y} = \mathbf{A}\underline{Y}$ is conventionally interpreted as meaning that $\underline{Y}$ is a *fixed point* of the linear transformation, **A**, which is not the intention in the present context.

It turns out that the intended definition **can** be presented in matrix-vector product form, but the correct matrix is not the one given in I.4. As an exercise, let's derive the correct matrix-vector product action for the operation defined by I.3. We first reformulate equation I.3 in matrix-vector form. Define an intermediate vector, $\underline{Y}$:

$$\begin{aligned}\underline{Y} &= \underline{W} + \underline{o}\\ &= \mathbf{T}\underline{R} + \underline{m} + \underline{o} \quad ,\end{aligned}$$

using the quantities defined in I.1--I.2. (In the suggested block diagrams above we combine $\underline{m}$ & $\underline{o}$ into a single offset vector, but we leave them in separate form for now to facilitate comparison with the FCD text.) Compare the first component of $\underline{Y}$ with the first equation in I.3 to see that $Y'_0 = Y_0$.

The first recursive definition in I.3 is

$$Y'_1 = r_{10}Y'_0 + Y_1 \ ,$$

where $Y_1 = W_1+o_1$ as given by the definition of $\underline{Y}$ above. Thus, Y'_0 and Y'_1 are the first two entries in the matrix-vector product $\mathbf{R}^{(1)}\underline{Y}$, where

$$\mathbf{R}^{(1)}Y = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ r_{10} & 1 & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & & & \ddots \end{bmatrix} \begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} Y_0' \\ Y_1' \\ Y_2 \\ \vdots \end{pmatrix}$$

The next recursively defined entry is

$$Y'_2 = r_{20}Y'_0 + r_{21}Y'_1 + Y_2 \ ,$$

which is obtained by multiplying $\mathbf{R}^{(1)}\underline{Y}$ by the matrix $\mathbf{R}^{(2)}$ as follows:

$$\mathbf{R}^{(2)}\mathbf{R}^{(1)}Y = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ r_{20} & r_{21} & 1 & \cdots \\ \vdots & & & \ddots \end{bmatrix} \begin{pmatrix} Y_0' \\ Y_1' \\ Y_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} Y_0' \\ Y_1' \\ Y_2' \\ \vdots \end{pmatrix}$$

Continuing in this fashion, the *i*th recursively defined entry, Y'_i , is obtained as the *i*th entry of the iterated matrix-vector product $\mathbf{R}^{(i)\cdots}\mathbf{R}^{(2)}\mathbf{R}^{(1)}\underline{Y}$, where $\mathbf{R}^{(i)}$ has 1's on the diagonal, the recursion coefficients for Y'_i on the *i*th line, and zeros elsewhere. The complete recursion for the inverse dependency transform is therefore given by

$$\underline{Y}' = \mathbf{R}^{(Q-1)}\mathbf{R}^{(Q-2)\cdots}\mathbf{R}^{(2)}\mathbf{R}^{(1)}\underline{Y} = \mathbf{R}'\underline{Y} \ .$$

This is the correct matrix-vector product expression for the inverse dependency transform defined by I.3. Note first that this expression is not recursive in the vector domain; i.e., it has $\underline{Y}$ as the input vector and $\underline{Y}$' for output. Also, note that the matrix **R'** in this expression is **not** the same as the matrix **R** that appears in I.4. Indeed, **R'** has the form

$$\mathbf{R}' = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ r'_{10} & 1 & 0 & \cdots \\ r'_{20} & r'_{21} & 1 & \cdots \\ \vdots & & & \ddots \end{bmatrix}$$

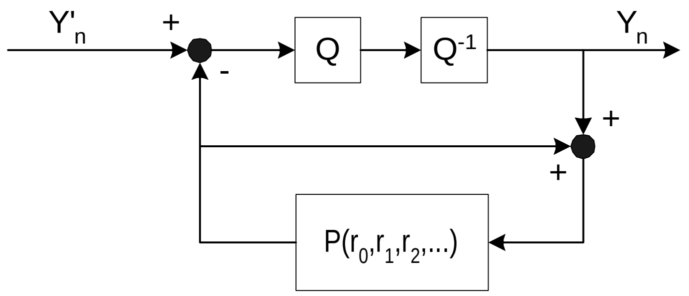
where the coefficients r'$_{ik}$ are determined by the matrix product $\mathbf{R'} = \mathbf{R}^{(Q-1)\cdots}\mathbf{R}^{(2)}\mathbf{R}^{(1)}$.

The above procedure is precisely the same as unrolling the inverse prediction loop following the quantization decoder in a DPCM decoder (though there is no quantization decoder in the present context) and writing it as a high-dimensional linear transform rather than as a recursive process. This is of conceptual/theoretical value since it means that this process can be written as a (nonrecursive) linear system, **R'**, with vector $\underline{Y}$ as input and vector $\underline{Y}$' as output. It is less clear, however, whether the nonrecursive realization of this system is appropriate in a practical implementation, or whether the nonrecursive (r'$_{ik}$) coefficients are more convenient to signal in the codestream. Since the decorrelation transform allows completely arbitrary linear transforms implemented in a nonrecursive fashion (matrix-vector product or FIR wavelet filter bank), it makes most sense to signal and implement the dependency removal transform in recursive form. This is most convenient for, e.g., autoregressive pixel modeling and prediction.

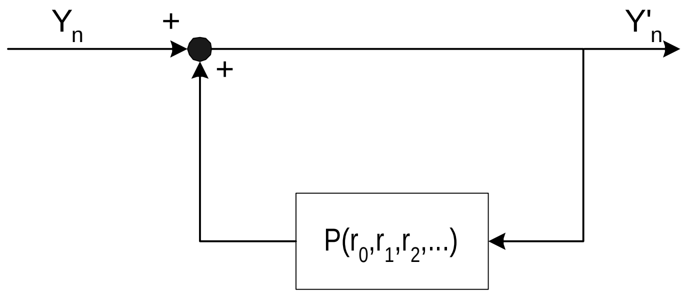
Recommendation. To keep the dependency removal transforms in recursive form, the misleading matrix-vector product in equation I.4 should be removed, and the recursive coefficients should not be referred to as "matrix" coefficients since the word "matrix" commonly implies "nonrecursive realization of a linear transform." Similarly, it would be preferable not to refer to the additive offset coefficients as "matrix" coefficients, since an additive offset is not even a linear transformation. A more generic approach that avoids implying anything about nonrecursiveness or linearity would be to refer to "arrays" of coefficients. Then the array of coefficients in an MCT marker segment for a decorrelation transform would be identical to the matrix representation of the transform while the array of coefficients for a dependency or additive offset transform would contain the recursion (resp., offset) coefficients.

Equation I.3, which defines dependency transforms, can be simplified by introducing another intermediate variable such as $\underline{Y} = \underline{W} + \underline{o}$, as shown above. With this definition of $\underline{Y}$, equation I.4 should be replaced by block diagrams for both forward and inverse dependency transforms, as shown below.

Forward dependency removal transform



Inverse dependency transform


Y_n
+
+
Y'_n
$P(r_0,r_1,r_2,...)$